Ratkaisumallien historia

Jaakko Vuolasto

Helsinki 25.1.2001

Seminaariesitelmä

HELSINGIN YLIOPISTO

Tietojenkäsittelytieteen laitos

Sisällys

1 Johdanto

Ratkaisumalli (*Design Pattern*) on käytännössä hyväksi todettu tapa ratkaista jokin tietty ongelma. Tämän esityksen tavoitteena on kertoa ratkaisumallin käsitteen lyhyehkö historia – miten se syntyi arkkitehtuurin kontekstissa, miten se myöhemmin omaksuttiin ohjelmistotuotantoon ja edelleen miten se saavutti nykyisen asemansa olennaisena osana erityisesti olioparadigmaa.

Esitelmän luku 2 esittelee ratkaisumallin yleisellä tasolla sekä esimerkin avulla. Varsinaiseen historiaan paneudutaan luvusta 3 alkaen: se käsittelee amerikkalaista arkkitehtia Christopher Alexanderia ja hänen ajatuksiaan. Ratkaisumallien alkuaikoja ohjelmistotuotannossa käydään läpi luvussa 4. Lähestymistapa on melko henkilökeskeinen. Luku 5 on lyhyt yhteenveto.

Historiaosuus noudattelee pääpiirteittäin ytimekästä WWW-esitystä [HOP], joka on erittäin hyvä johdanto aiheeseen. Christopher Alexander –osuus perustuu Nikos A. Salingarosin [Sal00] ja Doug Lean [Lea94] artikkeleihin.

2 Ratkaisumallin käsite ohjelmistotuotannossa

Ratkaisumalli muodostuu asiayhteydestä, ongelmasta ja ratkaisusta [Bus96]:

- Jokainen ratkaisumalli liittyy aina johonkin tiettyyn **asiayhteyteen.** Asiayhteydellä tarkoitetaan käytännössä tilannetta ja ympäristöä, jossa ongelma esiintyy.
- Ratkaisumallin esittämä **ongelma** esiintyy toistuvasti. Ongelma voidaan määritellä joukkona jännitteitä (*forces*).
- **Ratkaisu** kuvaa suunnitteluun kuuluvat elementit ja niiden väliset suhteet, eri elementtien vastuut sekä yhteistoiminnan. Ratkaisu on kokoonpano, joka tasapainottaa jännitteet.

Niin sanottu *Gang of Four* –ryhmä (Erich Gamma, Richard Helm, Ralph Johnson, John Vlissides) painottaa kirjassaan [Gam95] yllä mainittujen ominaisuuksien lisäksi ratkaisumallin nimeä ja seurauksia:

- Ratkaisumallin **nimi** toimii tunnisteena ja viestinnän välineenä; se kasvattaa ohjelmistokehittäjien yhteistä sanastoa.

- Ratkaisumallin soveltamisella on aina tiettyjä **seurauksia**. Mallissa on etuja ja haittoja, ja niitä on syytä punnita kun mallia käytetään.

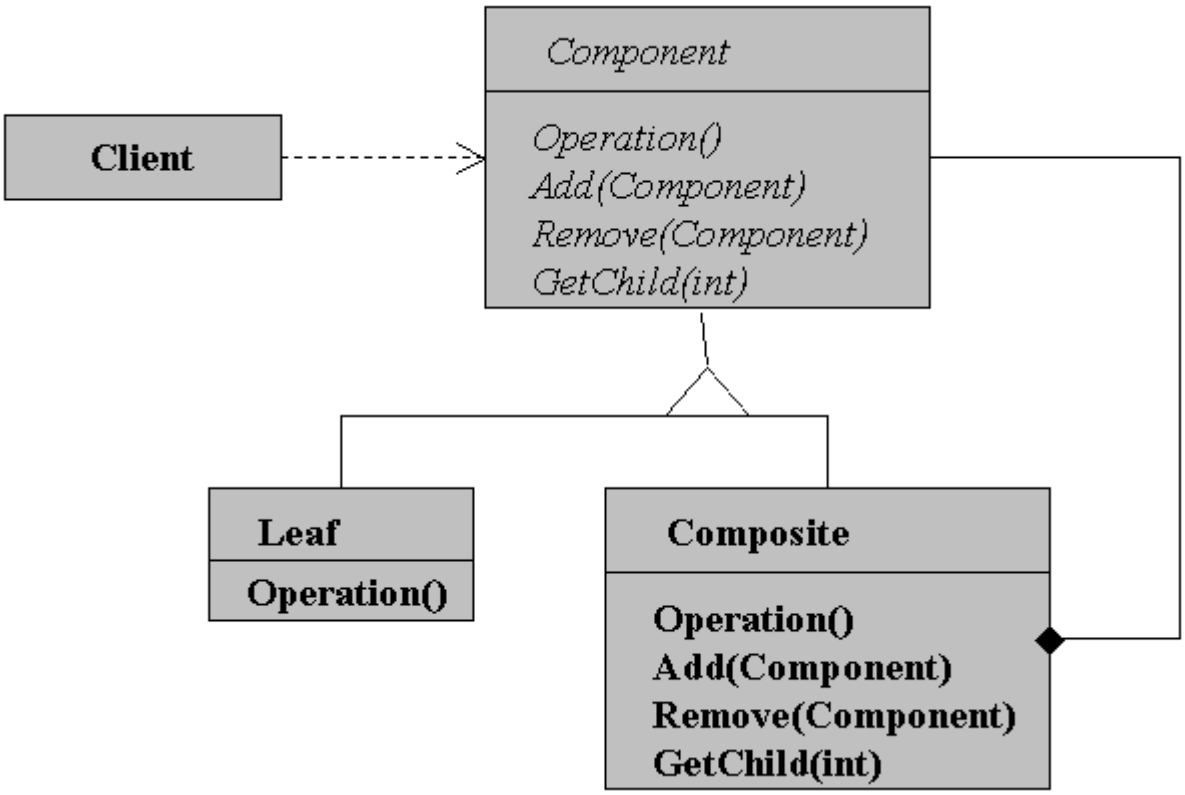
Sekä Gamman ja kumppaneiden kirja [Gam95] että Buschmannin ja kumppaneiden kirja [Bus96] ovat esimerkkejä ratkaisumalliluetteloista (*Pattern Catalogs*) – jopa jonkinlaisen de facto –standardin aseman saavuttaneista sellaisista – eli hakuteoksista, jotka listaavat useita yksittäisiä malleja. Luettelot eivät välttämättä ota kantaa, miten malleja voi soveltaa suurempien kokonaisuuksien (arkkitehtuurit, sovellukset, ohjelmistot) suunnittelussa.

Uusia ratkaisumalleja ei yleensä keksitä, vaan ne löydetään olemassa olevista ohjelmistoista. Silti malli ei ole yhtä kuin valmis koodinpätkä, jonka voi leikata ja liimata lähdekooditiedostosta toiseen. Toki ratkaisumalliin voi liittyä esimerkkitoteutus jollakin ohjelmointikielellä. Ytimeltään ratkaisumalli on kuitenkin aina abstrakti – kysymyksessä on uudelleenkäyttö suunnittelun, ei toteutuksen tasolla.

2.1 Esimerkki: Composite

Composite-ratkaisumalli soveltuu hierarkioiden esittämiseen, erityisesti kun jotakin rakennetta halutaan käsitellä tekemättä eroa rakenteisten ja yksinkertaisten olioiden välillä [Gam95]. Mallin ytimenä on abstrakti yliluokka, joka esittää sekä rakenteista että yksinkertaista oliota. Yliluokka esittelee molemmille tyypeille yhteiset operaatiot, sekä operaatiot, joita rakenteinen olio tarvitsee.

Tässä mallissa asiakas (*Client*) käyttää *Component*-luokan määrittelemää rajapintaa rakenteen sisältämien olioiden käsittelyyn. Jos pyynnön vastaanottava olio on tyypiltään *Leaf*, se toteuttaa halutun operaation suoraan itse. *Composite*-luokan ilmentymä puolestaan yleensä välittää pyynnön edelleen lapsilleen.



Kuva 1. Composite-ratkaisumallin luokkakaavio.

Composite on ollut mukana luettelossa [Gam95] sen alkuvaiheista lähtien [HOP]. Mallia on sovellettu muun muassa monissa käyttöliittymäkirjastoissa tai –kehyksissä[1].

3 Christopher Alexander

Ratkaisumallin juuret ovat amerikkalaisen arkkitehdin Christopher Alexanderin kirjoituksissa. Alexander syntyi Wienissä vuonna 1936, vietti lapsuutensa Englannissa ja valmistui Cambridgen yliopistosta opiskeltuaan matematiikkaa ja arkkitehtuuria. Tohtoriksi Alexander väitteli Harvardin yliopistossa 1963 [Sal00]. Väitöskirja julkaistiin myöhemmin nimellä *Notes on the Synthesis of Form* [Ale64][2]. Teoksen perusajatus on, että nykyaikainen arkkitehtuuri on epäonnistunut. Sen suunnittelumenetelmät ja käytännöt ovat virheellisiä, niiden tuloksena syntyy

[1] Esimerkiksi ET++, InterViews. Composite-mallia on hyödynnetty myös RTL Smalltalk-kääntäjän sovelluskehyksessä. Ks. [Gam95], sivu 172.

[2] Alexanderin teokset ovat mukana lähdeluettelossa, mutta eivät varsinaisina lähteinä. Valitettavasti aika ei riittänyt niiden lukemiseen. Luku 3 perustuu teosten kommentaareihin.

tuotteita, joihin käyttäjät eivät ole tyytyväisiä [Lea94]. Tällainen teesi ei liene vieras ohjelmistotuotannossakaan.

Alexander on toiminut Berkeleyn yliopiston arkkitehtuurin professorina vuodesta 1963 sekä *Institute of Environmental Structure* –laitoksen johtajana. Hänet kutsuttiin Ruotsin kuninkaallisen akatemian[3] jäseneksi 1980 ja *American Academy of Arts and Sciences* jäseneksi 1996.

Kuva 2. Christopher Alexander.

Alexanderin ajatukset ovat saavuttaneet suosiota arkkitehtuurin – ja tietojenkäsittelytieteen sekä ohjelmistotuotannon – ohella muillakin aloilla: malleja on sovellettu tai aiotaan soveltaa ainakin organisaatioiden suunnittelussa. Mallin käsitteellä on myös filosofinen ulottuvuus: Alexanderin näkemykset edustavat taoistista näkökulmaa [Sal00]. Tämä tulee esille ensimmäisen kerran kirjassa *The Timeless Way of Building* [Ale79]. Teosta voi pitää filosofiseen tyylin kirjoitettuna kirjana arkkitehtuurista tai toisaalta filosofisena pohdiskeluna, joka ottaa esimerkkinsä arkkitehtuurin alueelta.

The Timeless Way of Building –teoksen eräänlainen pari on *A Pattern Language* [Ale77]: edellinen esittelee arkkitehtuuria mullistavia periaatteita, esimerkiksi yleisiä malleja (*patterns*) tilasta (*space*) tai tapahtumista (*events*). Jälkimmäinen sisältää näiden periaatteiden käytännön

[3] Swedish Royal Academy.

yksityiskohtia [Lea94]: kokoelman malleja (253 kappaletta). Jokainen malli kuvataan viiden osan avulla: nimen, esimerkin, kontekstin, ongelman ja ratkaisun.

4 Arkkitehtuurista ohjelmistotuotantoon

4.1 Beck ja Cunningham

Kent Beck ja Ward Cunningham olivat vuonna 1987 mukana viimeistelemässä suunnitteluvaihetta ja päättivät soveltaa Alexanderin oppeja käytännössä. He antoivat ohjelmiston käyttäjien edustajien osallistua työhön. Tuloksena oli viiden ratkaisumallin kokoelma: *WindowPerTask, FewPanes, StandardPanes, NounsAndVerbs, ShortMenus*.[4] Beck ja Cunningham esittelivät tulokset OOPSLA 87 -konferenssissa [Bec87]. Vastaanotto oli hyväksyvä, jopa innostunut, mutta asia jäi keskustelun tasolle.

4.2 James Coplien

Vuonna 1992 julkaistu *Advanced C++ Programming Styles and Idioms* [Cop92] on kokoelma C++-spesifisiä idiomeja, eli oikeastaan ratkaisumallien esimerkkitoteutuksia. Jim Coplien aloitti luettelon tekemisen 1980-luvun loppupuolella, ja kirjan varhaisia käsikirjoituksia käytettiin C++-kurssilla AT&T:lla [HOP].

Coplienin vaikutus ratkaisumalliyhteisössä ei rajoitu yhteen C++-spesifiseen teokseen. Hän oli mukana muun muassa *Hillside Group* –ryhmässä (ks. luku 4.5).

4.3 Bruce Anderson

Bruce Anderson oli puhujana Tools90-konferenssissa. Kuulijoiden joukossa istui muun muassa Erich Gamma. Myöhemmin keskustellessaan miehet totesivat olevansa samoilla linjoilla suhteessa uudelleenkäytettävien (olio-)ohjelmistojen tekemiseen [HOP]. Anderson puhui aiheesta uudelleen OOPSLA 92 –konferenssissa, tuloksena oli yhteenvedon tyyppinen *Towards an Architecture Handbook* [And92]. Anderson selitti lyhyesti, mitä ohjelmistoarkkitehtuurilla tarkoitetaan, mikä olisi tällaisen käsikirjan rooli ja mitä itse kirja tulisi sisältämään.

---

[4] Linkit näiden mallien kuvaukseen löytyvät lähteestä [HOP].

4.4 Coad ja Mayfield

Peter Coad ja Mark Mayfield vetivät keskusteluryhmän ratkaisumalleista OOPSLA 92 –konferenssissa [CoM92]. Keskustelua ohjasi joukko kysymyksiä: mitä ratkaisumallit ovat, miten niitä voi oppia ymmärtämään paremmin, miten malleja löydetään ja kuvataan, millaisia esimerkkejä on saatavilla, sekä mikä on mallien mahdollinen vaikutus ohjelmistotuotantoon tulevaisuudessa. Näin jälkikäteen on tietysti mielenkiintoista lukea lähes kymmenen vuoden takaisista visioista: Coad ja kumppanit ennustivat, että mallien vaikutus tulee olemaan valtava. Mallit tarjoavat perustuksen, jonka varaan voidaan rakentaa johdonmukaisia ohjelmistoarkkitehtuureja. Lisäksi malli korkeamman abstraktiotason käsitteenä palvelee sekä tyyli- että pedagogisia päämääriä. Coad julkaisi myös artikkelin [Coa92] ratkaisumalleista 1992.

4.5 Hillside Group

Elokuussa 1993 joukko ratkaisumalleista kiinnostuneita (Ward Cunningham, Ralph Johnson, Ken Auer, Hal Hildebrand, Grady Booch, Kent Beck and Jim Coplien) kokoontui Coloradossa pohtimaan, miten yhdistää Alexanderin ajatukset, olio-ohjelmointi ja osallistujien kokemus. Ryhmän nimeksi valittiin Hillside Group[5]. Tämän ryhmän työn tuloksia ovat esimerkiksi *Pattern Languages of Program Design* –julkaisut [CoS95, VCK96, MRB97, HFR99].

4.6 Gang of Four

Luvussa 2 mainittu *Gang of Four* –ryhmä muodostui OOPSLA 92 –konferenssissa. Miehet olivat tavanneet tosin jo vuotta aikaisemmin, samassa tilaisuudessa [HOP].

Jo klassikoksi muodostunut *Design Patterns: Elements of Reusable Object-Oriented Software* julkaistiin juuri ennen OOPSLA 94 –konferenssia. Sitä myytiin konferenssissa 750 kappaletta, yli seitsemän kertaa enemmän kuin saman kustantajan kirjoja yleensä yhden konferenssin aikana. Journal of Object-Oriented Programming valitsi kirjan aihepiirinsä parhaaksi vuonna 1995 [HOP].

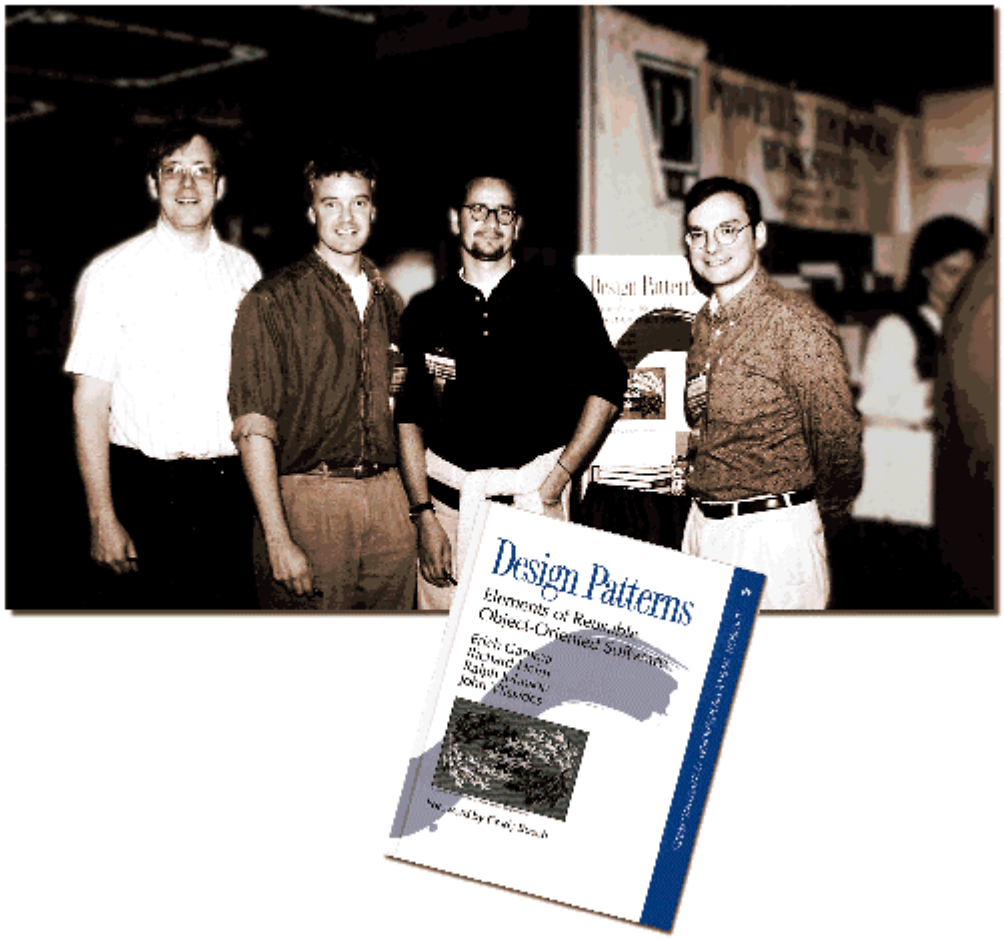
Kirja jaottelee ratkaisumallit kolmeen ryhmään niiden tarkoituksen perusteella: olioiden luomiseen, rakenteeseen ja käyttäytymiseen liittyvät mallit. Jokainen kategoria jakautuu edelleen

[5] Ks. http://c2.com/cgi-bin/wiki?HillsideGroup

luokka- ja ilmentymäkohtaisiin ratkaisumalleihin [Gam95]. Lopputuloksena on siis matriisi tai taulukko, jonka lokeroihin mallit näppärästi putoavat.

4.7 Frank Buschmann et al.

Kirjan [Gam95] ohella erittäin suuren suosion on saavuttanut Frank Buschmannin ja kumppaneiden *A System of Patterns – Pattern-Oriented Software Architecture* [Bus96]. Myös se jaottelee ratkaisumallit kolmeen eri ryhmään, tosin erilaisilla kriteereillä kuin Gamman kirja. Arkkitehtuurimallit (*Architectural Patterns*) kuvaavat järjestelmän perusrakennetta: jakoa alijärjestelmiin ja niiden keskinäistä työnjakoa. Ratkaisumallit (*Design Patterns*) kuvaavat (ali)järjestelmän rakennetta. Alimmalla tasolla idiomit (*Idioms*) ovat ohjelmointikielikohtaisia käytännön ratkaisuja [Bus96].



Kuva 3. Gang of Four ja Design Patterns. Vasemmalta oikealle: Ralph Johnson, Richard Helm, Erich Gamma ja John Vlissides

5 Ratkaisumallit tänään

Ratkaisumallit ovat vakiinnuttaneet asemansa osana olioparadigmaa. Tosin malli sinänsä ei ole sidottu pelkästään olio-ohjelmointiin, onhan olemassa esimerkiksi C- ja Lisp-idiomeja. Ratkaisumallit eivät välttämättä ole lunastaneet kaikkia odotuksia – tämä lienee selvää, sillä uuden tekniikan tai metodologian ympärillä on aina tyhjää puhetta ja mainoslauseita. Vakiintuneesta asemasta kertoo kuitenkin, että mallit ovat mukana oman laitoksemme opetusohjelmassa, esimerkiksi Ohjelmistoarkkitehtuurit-kurssilla [Lai00].

Tulevaisuuden kannalta olennaisimpia kysymyksiä on erilaisten ohjelmistotyökalujen tarjoama tuki ratkaisumallien hyödyntämiseen. Aiheesta on melko runsaasti tutkimusta, mutta valmiita tuotantokäyttöön soveltuvia työkaluja ei vielä välttämättä ole saatavilla [Vuo00]. Tilanteen korjautuminen vaatinee ajan lisäksi myös, että *Unified Modeling Language* [BRC99] ottaa selkeästi kantaa ratkaisumallien kuvaamiseen. De facto –standardin asemassa oleva kuvauskieli ei nykyisellään kykene täysin aukottomasti esittämään ratkaisumalleja [Lai00].

Vaikka ratkaisumalli käsitteenä ei ole kovin vanha, se on jo ehtinyt levitä arkkitehtuurista ohjelmistotuotantoon. Sinänsä Alexanderin ajattelu ja mallikäsitteen synty on lohdullista luettavaa ohjelmistoalan työntekijöille ja tutkijoille: muualtakin löytyy huonoja, vaillinaisia menetelmiä ja käytäntöjä.

Lähteet

And92 Anderson B., Towards an Architecture Handbook. Poster Submission, addendum to the proc. of OOPSLA92, Vancouver 1992.

BeC87 Beck K., Cunningam W., Using Pattern Languages for Object-Oriented Programs. Technical Report No. CR-87-43, September 1987. <URL:http://c2.com/doc/oopsla87.html>. Viimeksi tarkistettu 16.1.2001.

Bus96 Buschmann F., Meunier R., Rohnert H., Sommerlad P., Stal M., A System of Patterns – Pattern-Oriented Software Architecture. John Wiley & Sons, 1996.

CoM92 Coad P., Mayfield M., Patterns. Workshop Report, addendum to the proc. of OOPSLA92, Vancouver 1992.

Gam95 Gamma E., Helm R., Johnson R., Vlissides J., Design Patterns – Elements of Reusable Object-Oriented Software. Addison-Wesley, 1995.

HOP History Of Patterns. <URL:http://c2.com/cgi-bin/wiki?HistoryOfPatterns>. Viimeksi tarkistettu 16.1.2001.

Lai00 Laine H., Ohjelmistoarkkitehtuurit. Luentomoniste, Helsingin yliopiston tietojenkäsittelytieteen laitos, 2000.

Lea94 Lea D., Christopher Alexander: An Introduction for Object-Oriented Designers. ACM SIGSOFT Software Engineering Notes 19, 1, 1994, 39-46. <URL:http://gee.cs.oswego.edu/dl/ca/ca/ca.html>. Viimeksi tarkistettu 23.1.2001.

Sal00 Salingaros N., Some Notes on Christopher Alexander. <URL:http://www.math.utsa.edu/sphere/salingar/Chris.text.html>. Viimeksi tarkistettu 16.1.2001.

Muita viitteitä

Ale64 Alexander C., Notes on the Synthesis of Form. Harvard University Press, 1964.

Ale77 Alexander C., A Pattern Language. Oxford University Press, 1977.

Ale79 Alexander C., The Timeless Way of Building. Oxford University Press, 1979.

BRC99 Booch R., Rumbaugh J., Jacobson I., The Unified Modeling Language User Guide. Addison-Wesley, 1999.

Coa92 Coad P., Object-Oriented Patterns. Communications of the ACM 35, 9, 1992, 152-159.

Cop92 Coplien J., Advanced C++ Programming Styles and Idioms. Addison-Wesley, 1992.

CoS95 Coplien J., Schmidt D. Pattern Languages of Program Design, Addison-Wesley, 1995.

EgG92 Eggenschwiler T., Gamma E., ET++SwapsManager: Using Object Technology in the Financial Engineering Domain. ACM SIGPLAN Notices 27, 10, 1992, 166-177.

HFR99 Harrison N., Foote B., Rohnert H. Pattern Languages of Program Design 4, Addison-Wesley, 1999.

MRB97 Martin R., Riehle D., Buschmann F. Pattern Languages of Program Design 3, Addison-Wesley, 1997.

VCK96 Vlissides J., Coplien J., Kerth N. Pattern Languages of Program Design 2, Addison-Wesley, 1996.

Vuo00 Vuolasto J., Ratkaisumallien hyväksikäyttö ohjelmistotyökaluissa. Esitelmä Ohjelmistotuotantovälineet-seminaarissa, Helsingin yliopiston tietojenkäsittelytieteen laitos, 2000. <URL:http://www.cs.helsinki.fi/u/jovuolas/se-tools/vuolasto.pdf>.

Kuvien URL-osoitteet

Kuva 2 http://mather.ar.utexas.edu/Resources/center/symposia/good_building/ participants.html

Kuva 3 http://www.ddj.com/articles/1998/9803/9803a/9803a.htm